An Optimal Task Management and Control Scheme for Military Operations with Dynamic Game Strategy

Tao Zhang, Chengchao Li, Dongying Ma, Xiaodong Wang, Chaoyong Li

Abstract

As is well known, military operation in a combat scenario is extremely intricate and often prone to optimal and real-time decisions. In this paper, we study task management and control problem for military operations with a dynamic game strategy. Toward this, the underlying problem is modeled by a matrix game scheme with performance index defined for both parties. Then, we proceed to present a fast and optimal search algorithm, inspired by graph theory and Kuhn-Munkres algorithm, to solve dimension explosion problem inherent with matrix game scheme and retrieve the optimal solution for each combat entity. Simulation results verified the effectiveness of the proposed scheme.

Index Terms

Game theory, air combat, task management, search algorithm

I. Introduction

Military operations in a combat scenario is an extremely complicated system involving multiple level of decision making with conflict and even competitive objectives. In general, the command center of each side should be able to seek a feasible, or optimal if possible, solution to allocate the limited resources by taking into account the battlefield situation and possible adversary actions, in order to optimize the performance index of the entire team [1–7]. In [5], a UCAV combat task assignment model is established considering air superiority and attack income to meet the needs of different tactics. [7] investigates the task assignment problem where the value of the targets is time-varying, and combines the dynamic programming and multisubgroup ant colony algorithm. However, the challenge of the underlying problem is not only how to model the combat operation systematically, but also how to seek an optimal and real-time solution for an otherwise computational exhaustive problem.

Dynamic game theory has received much attention for its advantages of considering multi-player competition in complex situations and formulating feasible strategies for agents [8–14]. Cruz and Simaan made a breakthrough on application of game theory in control of military operations [2–4], but finding a Nash equilibrium in a two-player game has been proved to be a PPAD-Complete problem [15]. In [3], a suboptimal solution over a small moving time horizon instead of the entire time horizon for computing the game solutions is proposed. [16] and [17] discussed an efficient method for approximating the reaction strategy for one team given a strategy chosen by another team using a unit level team resource allocation algorithm (ULTRA) in competitive multi-team target assignment problems. Duan [18] applied a hybrid predator-prey particle swarm optimization (PP-PSO) to the solution of the mixed strategy Nash solution, which can effectively solve the dynamic task allocation problem of multi-agents. Orafa [19] introduced a Q-Learning algorithm in large state-action domains.

It should be pointed out that, albeit their effectiveness, most of aforementioned results struggles to produce real-time decisions against large scale operations, and may not be able to achieve a tradeoff between algebraic complexity and accuracy. In this paper, we attempt to tackle this problem using a maximum weight matching algorithm. In particular, the underlying task management problem is abstracted as the bipartite graph weight matching problem, and a classic maximum weight matching solution is thus introduced to reduce the computational burden without sacrificing the accuracy. Comparing with existing schemes, we proved that the proposed scheme retains the Nash equilibrium of the overall problem, as it should, while exhibiting superior performance in terms of convergence rate.

II. Preliminaries and problem formulation

A. Dynamic model of military operations

In this paper, we consider combat operations between two groups of entities, labeled as *Blue* and *Red*, with *Blue* defined as the attacking force and *Red* as the defending force. The number of combat units of *Blue* and *Red* are denoted by N^B and N^R, respectively, and the battlefield is limited to an area measured by $W \times L \times H$. In addition, each combat unit is assumed to equip a number of w weapons and its physical condition is measured by the health point (HP) $0 \leq p \leq 1$.

Tao, Chengchao, and Chaoyong are with College of Electrical Engineering, Zhejiang University, Hangzhou, China, 310027. Corresponding to: Chaoyong@zju.edu.cn

Dongying and Xiaodong are with Beijing Institute of Electronic System Engineering, Beijing, 100854, China.

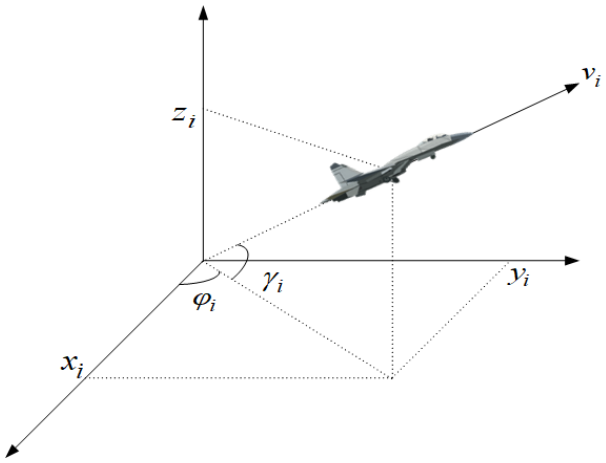


Fig. 1: Engagement geometry

Without loss of any generality, the combat operation is assumed to be evolved in a discrete time with time steps $t_k = kT$, $k = 0, ..., K$ and T is the predefined time interval. As depicted in Figure 1, dynamics of the ith unit can be described as follows,

$$\begin{aligned} x_i(k+1) &= x_i(k) + Tv_i(k)\cos\varphi_i(k)\cos\gamma_i(k) \\ y_i(k+1) &= y_i(k) + Tv_i(k)\sin\varphi_i(k)\cos\gamma_i(k) \\ z_i(k+1) &= z_i(k) + Tv_i(k)\sin\gamma_i(k) \\ v_i(k+1) &= v_i(k) + V_i(k) \\ \varphi_i(k+1) &= \varphi_i(k) + \Psi_i(k) \\ \gamma_i(k+1) &= \gamma_i(k) + \Upsilon_i(k) \end{aligned} \tag{1}$$

where $[x_i(k), y_i(k), z_i(k)]^T$ denotes the location, $v_i(k)$, $\varphi_i(k)$, and $\gamma_i(k)$ are the speed, flight path angle and heading angle during time interval T, respectively; $V_i(k) \in \{0, \pm V\}$, $\Psi_i(k) \in \{0, \pm\Psi\}$,$\Upsilon_i(k) \in \{0, \pm\Upsilon\}$ and V, Ψ, Υ are maximal allowed deviations for speed, flight path angle and heading angle, respectively.

Hence, the state of the ith unit is

$$\lambda_i^X(k) = [x_i^X(k), y_i^X(k), z_i^X(k), p_i^X(k), w_i^X(k)]^T \tag{2}$$

where $p_i^X(k)$ and $w_i^X(k)$ denote the remaining HP and numbers of weapons, respectively, and X denotes either the *Blue* force or *Red* force.

In addition, the number of weapons is updated by

$$w_i^X(k+1) = w_i^X(k) - \sum_{j=1}^{N^Y} c_{ji}^{YX}(k) \tag{3}$$

where $c_{ji}^{YX}(k) = 0, 1, ..., w_i^X(k)$ is the salvo size of unit X_i while attacking unit Y_i, with Y defined analogously as X.

The HP is updated as follows

$$p_i^X(k+1) = p_i^X(k) - \Delta p_i^X(k) \tag{4}$$

with

$$\Delta p_i^X(k) = \sum_{j=1}^{N^Y} P_{ij}^{XY}(k) Q_{ij}^{XY}(k) \tag{5}$$

where the attrition factor $P_{ij}^{XY}(k)$ represents the real kill rate of unit Y_j attacking unit X_i, and can be expressed as $P_{ij}^{XY}(k) = 1 - (1 - \beta_w PK_{ij}^{XY})^{s_j^Y}$, as $0 \le \beta_w \le 1$ defined as the weather factor that can potentially implicate the kill rate and PK_{ij}^{XY} the probability of kill under perfect weather condition. Subsequently, let s_j^Y be the average effective salvo size when unit Y_j fires at unit X_i, and

$$s_j^Y = c_{ij}^{XY}(k) \left(\frac{p_j^Y(k)}{p_i^X(k)} \right)^{\omega} \tag{6}$$

where $0 \le \omega \le 1$ is commonly known as the Weiss parameter [20, 21].

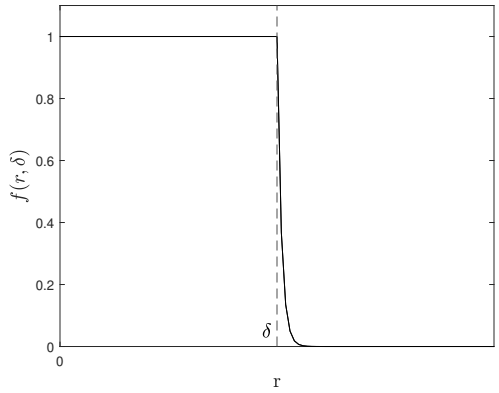


Fig. 2: An example of $f(r, \delta)$

Suppose $Q_{ij}^{XY}(k)$ denote the probability that a missile fired by Y_j effectively kills X_i, and [2]

$$Q_{ij}^{XY}(k) = \beta_{ij}^{XY}\left[1 - \exp\left(-\frac{p_j^Y(k)}{p_i^X(k)}\right)\right] \tag{7}$$

where $0 \leq \beta_{ij}^{XY} \leq 1$ is the probability that Y_j treats X_i as its target, and

$$\beta_{ij}^{XY} = f\left(\frac{g_i^X}{g_j^Y}, \delta\right) \tag{8}$$

where $0 < f(r, \delta) < 1$ is a monotonically decreasing function and $f(r, \delta) \to 1$ if $r \leq \delta$, $f(r, \delta) \to 0$ if otherwise, with g_i^X and g_j^Y defined as instantaneous load factor for X_i and Y_j, and δ as the maximal threshold allowed by the guidance command, as shown in Fig.2. That is, $f(r, \delta)$ will vanish exponentially should adversary's maneuverability exceeds its capability, as it should.

Hence, the control input for unit X_i is

$$u_i^X(k) = \left[V_i^X(k), \Psi_i^X(k), \Upsilon_i^X(k), c_i^X(k)\right]^T \tag{9}$$

where $c_i^X(k) = \left[c_{1i}^{YX}(k),\ ...,\ c_{N^Y i}^{YX}(k)\right]^T$ is the attack vector of X against Y, and $c_{ji}^{YX}(k) = 0$ if X_i doesn't choose unit Y_j as its target.

Remark: It is obvious that the proposed scheme combines motion planning, resource management, and target assignment into one holistic conjecture. With proper choice of u_i^X, unit X_i will not only be able to dynamically move along a prime direction, but also screen and acquire its next target with the highest probability of success.

B. Command constraint and payoff function

Unless otherwise specified, we consider the following constraints for all entities in the underlying scenario [2].

1) Target-selection constraint: At time instant k, no two combat units from the *Blue* force can attack (be targeted by) the same combat unit from the *Red* force, and vice versa. That is,

$$c_i^B(k)^T c_j^B(k) = 0, for\ each\ unit\ i \neq j\ of\ Blue \tag{10}$$

and

$$c_i^R(k)^T c_j^R(k) = 0, for\ each\ unit\ i \neq j\ of\ Red \tag{11}$$

2) Fire constraint: Let $d_{ij}^{XY}(k)$ and $\theta_{ij}^{XY}(k)$ be the distance and line-of-sight angle between X_i and Y_j at time instant k, respectfully. Then, X_i can only choose Y_j as its target of attack if

$$\begin{cases} d_{ij}^{XY}(k) \leq D_i^X \\ \theta_{ij}^{XY}(k) \leq \theta_i^X \end{cases} \tag{12}$$

where D_i^X is the maximal attack range for X_i and θ_i^X is the maximal steering angle for sensors attached to X_i, as illustrated in Figure 3.

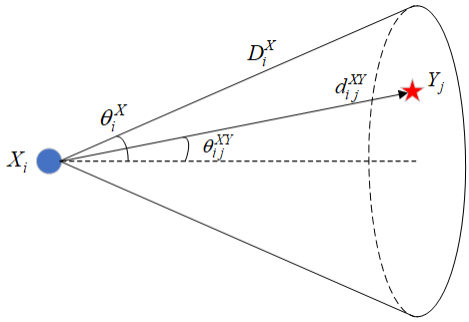


Fig. 3: Fire Constraint

3) Collision avoidance: Let D_{unit} and D_{ground} be the minimal safe distance between any pair of units and minimal safe distance from ground, respectively. Then, the following condition should be satisfied at any instant k, to avoid internal collision.

$$\begin{cases} d_{ij}(k) \geq D_{unit}, & \forall i \neq j \\ d_{i,ground}(k) \geq D_{ground} \end{cases} \tag{13}$$

where $d_{i,ground}$ is the distance between unit i and the ground/surface.

4) Maneuverability constraint: Let $v_i^{\max}$, $\varphi_i^{\max}$ and $\gamma_i^{\max}$ be the upper limit on speed, flight path angle and heading angle, respectively, for unit i and

$$\begin{cases} v_i(k) \leq v_i^{\max} \\ \varphi_i(k) \leq \varphi_i^{\max} \\ \gamma_i(k) \leq \gamma_i^{\max} \end{cases} \tag{14}$$

In this paper, the goal of the *Blue* force is to destroy the combat units of the *Red* force as many as possible, while preserving its own entities, and vice versa for the *Red* force. In other words, performance index for each side over the entire time horizon can be designed as [2]

$$J_{1,K}^X = Q^X\left(\Lambda_K^X, u_K^X, u_K^Y\right) + \sum_{k=1}^{K-1} R^X\left(\Lambda_k^X, u_k^X, u_k^Y\right) \tag{15}$$

where $Q^X(.)$ and $R^X(.)$ are terminal performance index function and integral performance index function about the state $\Lambda_k^X = \left[\lambda_1^X(k), ..., \lambda_{N^X}^X(k)\right]^T$ of X and the strategies of both sides, respectively.

However, it should be pointed out that searching an optimal/Nash solution to (15) over the entire time horizon is time-consuming and, if even possible, computational exhaustive [8], its algebraic complexity could be overwhelming as k evolves. Hence, it is a common practice to revoke receding horizon principle and introduce a one-step-ahead performance index for dynamic game strategy, as suggested in [3] and its performance in formation control can be found in [22]. In this case, the strategies u_k^X and u_k^Y will only influence the payoff function only at time instant $k+1$. As a result, performance index becomes

$$J_{k,k+1}^X = R^X\left(\Lambda_k^X, u_k^X, u_k^Y\right) \tag{16}$$

Here we define the payoff functions for both the *Blue* and *Red* forces as follows

$$J_{k,k+1}^B\left(u_k^B, u_k^R\right) = \sum_{i=1}^{N^B}\left(\alpha_i^B p_i^B(k) + \alpha_{tar}^R \Delta p_{tar}^R(k)\right) r_i^B \tag{17}$$

and

$$J_{k,k+1}^R\left(u_k^B, u_k^R\right) = \sum_{i=1}^{N^R}\left(\alpha_i^R p_i^R(k) + \alpha_{tar}^B \Delta p_{tar}^B(k)\right) r_i^R \tag{18}$$

where α_i^B and α_i^R are the weights to be specified, $\Delta p_{tar}^R(k)$ and $\Delta p_{tar}^B(k)$ are the loss of HP of the target of unit i, and r_i^B and r_i^R are designed to evaluate the motion control performance, and

$$r_i^B = \sum_{j=1}^{N^R} \alpha_j^R e^{-\hat{d}_{ij}(k)} \tag{19}$$

and

$$r_i^R = \sum_{j=1}^{N^B} \alpha_j^B e^{-\hat{d}_{ij}(k)} \tag{20}$$

where $\hat{d}_{ij}$ is the normalized distance between units i and j.

Therefore, the problem to be solved in this paper is, for both the *Blue* and *Red* forces, to find a Nash equilibrium $\left(u_k^{B*}, u_k^{R*}\right)$ that the value of $J_{k,k+1}^B$ and $J_{k,k+1}^R$ couldn't be increased further with unilateral decisions, that is,

$$J_{k,k+1}^B\left(u_k^{B*}, u_k^{R*}\right) \geq J_{k,k+1}^B\left(u_k^B, u_k^{R*}\right), \forall u_k^B \in U_k^B \tag{21}$$

$$J_{k,k+1}^R\left(u_k^{B*}, u_k^{R*}\right) \geq J_{k,k+1}^R\left(u_k^{B*}, u_k^R\right), \forall u_k^R \in U_k^R \tag{22}$$

where U_k^B and U_k^R are sets of all admissible strategies at time instant k for the *Blue* and *Red* forces, respectfully.

C. Weighted bipartite graph

Before proceeding further, we introduce some preliminary results on graph theory. A weighted graph can be captured by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$, where $\mathcal{V} = \{1, 2, ..., n\}$ is the set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\omega : \mathcal{E} \to \Re$ denotes the weight of the corresponding edge. In addition, a weighted graph is called bipartite graph (i.e., bigraph) if its vertices can be divided into two disjoint and independent sets, namely, $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_R$, and every edge connects a vertex in $\mathcal{V}_L$ to one in $\mathcal{V}_R$. A matching M is a set of the weighted edges chosen in such a way that no two edges share an endpoint, and the *maximum weight matching* (MWM) problem is to find a matching M^* such that the sum of weights of all edges in M^* is the largest among all matchings, that is [23]

$$M^* = \arg\max_{M \subset \mathcal{E}} \sum_{e \in M} \omega(e) \tag{23}$$

Moreover, an *augmenting path* for a matching M is a path with an odd number of edges $e_1, e_2, ..., e_m$ such that $e(odd)$ not in M and $e(even)$ in M. Let ε_i^L and ε_j^R denote the label of left vertex i and right vertex j in $\mathcal{G}$, then [24]

$$\varepsilon_i^L + \varepsilon_j^R \geq \omega_{ij}, \ \forall \ e_{ij} \in \mathcal{E} \tag{24}$$

where ω_{ij} is the weight of the edge e_{ij}.

A graph $\mathcal{G}_e = (\mathcal{V}, \mathcal{E}_e, \omega_e)$ is said to be an *equality graph* for graph $\mathcal{G}$ if

$$\mathcal{G}_e = \left\{\hat{e}_{ij} \in \mathcal{E} \,\middle|\, \varepsilon_i^L + \varepsilon_j^R = \hat{\omega}_{ij}\right\} \tag{25}$$

The following lemma summarizes the relation of the MWM between $\mathcal{G}_e$ and $\mathcal{G}$, and its proof is followed immediately.

Lemma 1. *If an equality graph $\mathcal{G}_e$ for graph $\mathcal{G}$ has a matching, M_e, then M_e is a maximum weight matching in $\mathcal{G}$.*

Proof. Let S be the sum of the weights of all edges in graph $\mathcal{G}_e$, that is,

$$\sum_{i=1}^{N^L} \varepsilon_i^L + \sum_{j=1}^{N^R} \varepsilon_j^R = \sum \hat{\omega}_{ij} = S \tag{26}$$

Assume that there is another graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \omega')$ that is not an equality graph with a matching M' such that

$$\sum_{i=1}^{N^L} \varepsilon_i^L + \sum_{j=1}^{N^R} \varepsilon_j^R = \sum {\omega'}_{ij} = S_1 > S \tag{27}$$

indicating that $\exists \omega'_{ij} > \hat{\omega}_{ij}$, which contradicts to the fact that $\varepsilon_i^L + \varepsilon_j^R = \hat{\omega}_{ij} \geq \omega'_{ij}$, and this concludes the proof. □

III. Main results

In this section, we attempt to retrieve the optimal solution for the underlying management and control problem with the *Action-Reaction Search* (i.e., ARS) algorithm along with Kuhn-Munkres algorithm (i.e., KM), and special attention will be paid on how to alleviate the dimension explosion problem associated with matrix game strategy without sacrificing performance. As is well established [2, 3], albeit its effectiveness, dynamic matrix game strategy is plagued by the dimension explosion problem, which will grow exponentially as the game evolves and could be cumbersome in real time decision. To be more exact, unit i has a number of $n_i^X(k)$ choices at time instant k, and

$$n_i^X(k) = m_i^X(k) c_{ji}^{YX}(k) \tag{28}$$

where $m_i^X(k)$ is the number of the available targets at time instant k. It follows that the number of choices for the whole force is, at time instant k

$$n^X(k) = \prod_{i=1}^{N^X} n_i^X(k) \tag{29}$$

Obvious that choice of control input will grow exponentially as k evolves, and eventually saturate computation capacity if no punitive action is expected. For instance, suppose the *Blue* force has 3 combat units that each equipped with 5 weapons, and there are 4 *Red* units available at the moment. Then, the number of choices for any *Blue* unit will be $n_i^B(k) = 567$, and the total number of choices for the *Blue* force will be $n^B(k) = 567^3$, any attempt to retrieve a Nash or Stackleberg solution will be computational exhaustive. Toward this, we apply the ARS algorithm to mitigate dimension explosion problem by avoiding calculation of the whole payoff matrix. Instead, each unit only needs to search its corresponding row/column to find an optimal solution, and its effectiveness in extracting pure Nash equilibrium is examined in [25]. The algorithmic procedure is summarized in Algorithm 1.

Algorithm 1 Action-Reaction Search Algorithm

OUTPUT: $\left(u_k^{B*}, u_k^{R*}\right)$
BEGIN:

1: Select a random strategy $\tilde{u}_k^R$ of *Red*
2: Set maximum iteration steps K
3: $i = 0$
4: Find $\hat{u}_k^B$ such that $J_{k,k+1}^B\left(\hat{u}_k^B, \tilde{u}_k^R\right) \geq J_{k,k+1}^B\left(u_k^B, \tilde{u}_k^R\right), \forall u_k^B \in U_k^B$
5: Let $\tilde{u}_k^B = \hat{u}_k^B$
6: Find $\hat{u}_k^R$ such that $J_{k,k+1}^R\left(\tilde{u}_k^B, \hat{u}_k^R\right) \geq J_{k,k+1}^R\left(\tilde{u}_k^B, u_k^R\right), \forall u_k^R \in U_k^R$
7: **if** $\hat{u}_k^R == \tilde{u}_k^R$ **then**
8: $\quad u_k^{B*} = \hat{u}_k^B$, $u_k^{R*} = \hat{u}_k^R$
9: $\quad$ **goto END**
10: **else**
11: $\quad$ **if** $i > K$ **then**
12: $\quad\quad$ Nash equilibrium not found.
13: $\quad\quad$ **goto END**
14: $\quad$ **else**
15: $\quad\quad \tilde{u}_k^R = \hat{u}_k^R$
16: $\quad\quad i = i + 1$
17: $\quad\quad$ **goto** Step 4
18: $\quad$ **end if**
19: **end if**

END

Lemma 2. *If Algorithm 1 can proceed to Step 8, then* $\left(\hat{u}_k^B, \hat{u}_k^R\right)$ *is a Nash equilibrium* $\left(u_k^{B*}, u_k^{R*}\right)$.

Proof. Since $\hat{u}_k^R = \tilde{u}_k^R$ and $\tilde{u}_k^B = \hat{u}_k^B$, in Step 5 and Step 7, then the equations in Step 4 and Step 6 can be rewritten as

$$J_{k,k+1}^B\left(\hat{u}_k^B, \hat{u}_k^R\right) \geq J_{k,k+1}^B\left(u_k^B, \hat{u}_k^R\right), \forall u_k^B \in U_k^B$$
$$J_{k,k+1}^R\left(\hat{u}_k^B, \hat{u}_k^R\right) \geq J_{k,k+1}^R\left(\hat{u}_k^B, u_k^R\right), \forall u_k^R \in U_k^R$$

which are consistent with the definition of the Nash equilibrium as in (21) and (22), and this concludes the proof. □

The ARS gradually finds the Nash equilibrium (if exists) through an optimal response strategy made by both players to each other's strategy. In particular, the ARS algorithm will maintain two arrays during the procedure, and those arrays will be used to store the searched indices of the both forces. In this case, if the current strategy has been marked, which means the search path is in a loop and no Nash equilibrium will be found.

As indicated in Algorithm 1, the ARS algorithm can locate an optimal solution with only one iteration in the best case scenario, however, it may have to search the whole matrix in the worst case, which could be time-consuming hence cumbersome for real-time decision. In what follows, we introduce a pruning method to ease the computational burden in Step 4 or Step 6 in Algorithm 1. Note that the pruning method is commonly adopted in data processing and neural network [26] to eliminate the unrealistic strategies through reduction of the dimension. Noted that dimension explosion of the underlying problem attributes in part to the intricate combination of decision factors on both forces, which naturally contains a series of unrealistic/inappropriate choices that should be eliminated through basic pruning principle.

For a single combat unit X_i at time instant k, let $U_k^{X_i} = \left\{u_1^{X_i}, u_2^{X_i}, ..., u_{n_i^X}^{X_i}\right\}$ be its candidate set of actions, with $u_p^{X_i} \in U_k^{X_i}, p = 1, ..., n_i^X$ termed as a specific action/control of unit X_i. Accordingly, the set of payoff function for unit X_i is $J_{k,k+1}^{X_i} = \left\{j_1^{X_i}, j_2^{X_i}, ..., j_{n_i^X}^{X_i}\right\}$. Then, a candidate action $u_l^{X_i} \in U_k^{X_i}, l = 1, ..., n_i^X$, can be eliminated if one of the following condition is satisfied

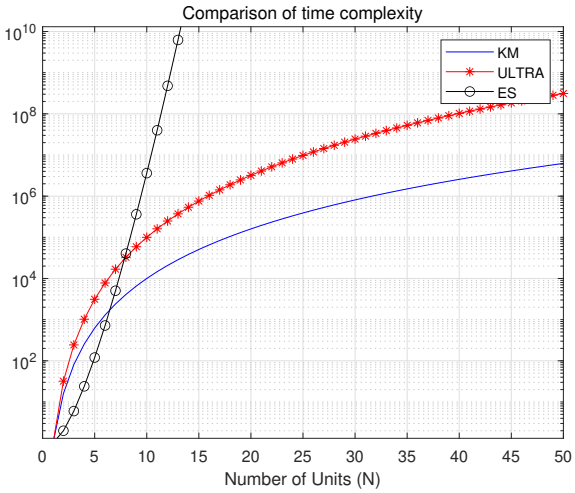


Fig. 4: Comparison of complexity (i.e., the degree of freedom in ULTRA is set to 2)

1) $u_l^{X_i}$ and $u_p^{X_i}$ has the same target, and $j_l^{X_i} \leq j_p^{X_i}$ for $l, p = 1, ..., n_i^X$.
2) both $u_l^{X_i}$ and $u_p^{X_i}$ attack no target, and $j_l^{X_i} \leq j_p^{X_i}$ for $l, p = 1, ..., n_i^X$.

It is implied that condition 1 is proposed to avoid repeated selections of the same target, while condition 2 ensures that selection of any particular target will produce a better payoff than no selection at all.

In what follows, we demonstrate that the underlying task management problem can be captured by a bigraph, and as such, fundamental principles in bigraph theory can be readily applied in searching for the optimal response strategy. More specifically, for a reduced choice set $U_k^{X_i}$ for unit X_i, choice $u_j^{X_i} \in U_k^{X_i}$ can be represented by a tuple (i, j), where i represents the unit X_i, and j represents the target (i.e., unit Y_i), and $j = 0$ if unit X_i opts out of attack. Therefore, in order to search for an optimal response strategy for X, all units in X can be regarded as vertices on one side of the bigraph, while units in Y along with an empty set (i.e., no attacking target) can be regarded as the vertices on the other side of the bigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$. Consequently, it can be concluded that, if an edge $e_{ij} \in \mathcal{E}$ exists in bigraph $\mathcal{G}$, there is a choice/option that unit X_i can attack Y_j, and the weight of the edge, ω_{ij}, is the payoff of the choice. The proceeding conclusion is applicable in task management problem in general, but for the underlying combat scenario, each left vertex should match a right vertex, but not necessarily vice versa, and by previous defined engagement principle, a right vertex can not be matched by multiple left vertices, with an exception for the empty set. In other words, the underlying problem can be converted into a maximum weight matching problem in a bipartite graph [27], which, as is well established, can be solved by Kuhn-Munkres (i.e., KM) algorithm [28–30].

In particular, KM algorithm attempts to find the maximum matching by searching the augmenting paths, and if an augmenting path can not be found, it modifies the labels of the vertices on the search path $\mathcal{V}^S = \mathcal{V}_L^S \cup \mathcal{V}_R^S$, and attempts again until the bigraph becomes an equality graph. The detailed algorithmic procedure can be summarized in Algorithm 2.

It is apparent that the algorithm complexity of the KM algorithm is

$$O\left(|\mathcal{E}|^2\right) = O\left(\left(N^X N^Y\right)^2\right) \tag{30}$$

where N^X refers to the number of units for the X force.

In the case of $N^X = N^Y = N$, its complexity is $O\left(N^4\right)$, while, in this case, the algorithm complexity for exhaustive search (i.e., ES) and the ULTRA [17] are

$$\begin{aligned} O_{ES} &= O\left(N!\right) \\ O_{ULTRA} &= O\left(N^{2\xi+1}\right) \end{aligned} \tag{31}$$

where ξ is the degree of freedom in [16].

A comparison of the complexity for the proposed KM algorithm, ES algorithm, and ULTRA algorithm is plotted in Figure 4, and it is clear that ES is indeed a better solution if the combat units are limited (i.e., $N < 6$), but KM becomes the apparent choice if the operation is extensive, which will be further classified in the following section.

IV. Simulation Results and Discussions

In this section, numerical simulations are conducted to validate the effectiveness of the proposed KM algorithm. For full disclosure, all simulations are performed on a desktop computer with a 3.6GHz AMD Ryzen 3500X CPU. Without loss of any generality, we consider an operation taking place on a $800 \times 700 \times 20$ (kilometers) region, and there are 7 *Blue* units with the mission to destroy at least 40% of a Fixed Target (FT) defended by 7 *Red* units. The initial conditions and probability of kill

Algorithm 2 Maximum Weight Matching

INPUT: Given opponent's strategy u_k^Y
OUTPUT: Optimal response $\hat{u}_k^X$

for each unit i **in** X **do**
    Find choice set $U_k^{X_i}$ for unit i
    Eliminate unnecessary choices from $U_k^{X_i}$
end for
Let left vertices be units of X, right vertices the units of Y and an additional empty unit
for each left vertex $i \in \mathcal{V}_L$ and right vertex $j \in \mathcal{V}_R$ **do**
    $\varepsilon_i^L = \max_{j \in \mathcal{V}_L} \{\omega_{ij}\}$, $\varepsilon_j^R = 0$
end for
Initialize the set of left vertices that match the right vertices $matched = []$
for each left vertex $i \in \mathcal{V}_L$ **do**
    while true **do**
        Set $\mathcal{V}_L^S$ and $\mathcal{V}_R^S$ to empty sets
        $found$=HUNGARIAN(i)
        if $found = true$ **then**
            break
        end if
        $\Delta = \min_{i \in \mathcal{V}_L^S} \min_{j \in \mathcal{V}_R^S} \left\{\varepsilon_i^L + \varepsilon_j^R - \omega_{ij}\right\}$
        for each left vertex $i \in \mathcal{V}_L^S$ **do**
            $\varepsilon_i^L = \varepsilon_i^L - \Delta$
        end for
        for each right vertex $j \in \mathcal{V}_R^S$ **do**
            $\varepsilon_i^R = \varepsilon_i^R + \Delta$
        end for
    end while
end for
Get matched result $\hat{u}_k^X$ from $matched$

function HUNGARIAN(left vertex i)
    Add left vertex i to $\mathcal{V}_L^S$
    for each right vertex j **in** right vertices **do**
        if $\varepsilon_i^L + \varepsilon_j^R = \omega_{ij}$ **then**
            if vertex j is an empty unit **then**
                return $true$
            end if
            if $j \notin \mathcal{V}_R^S$ **then**
                Add right vertex j to $\mathcal{V}_R^S$
                Let left vertex $m = matched[j]$
                if m is $null$ **or** HUNGARIAN(m)=$true$ **then**
                    $matched[j] = i$
                    return $true$
                end if
            end if
        end if
    end for
    return $false$
end function

for each unit are summarized in Table I and Table II. In particular, the *Blue* units consists of 5 weasels (BW) each equipped with 4 air-to-air missiles and 2 bombers (BB) each equipped with 5 air-to-ground bombs, while the *Red* units consists of 7 weasels (RD) each equipped with 3 air-to-air missiles and an air base (FT) with 4 ground-to-air missiles, the weighting coefficients of each unit are summarized in Table III. It is made clear that destroying the FT as well as preserving the bombers are the priority for the *Blue* force, while protecting the air base and destroying *Blue* bombers are priority for the *Red* force.

The sampling interval T is set to 1 second, and the maximal speed is chosen as 0.8 Mach for the *Blue* units and 1 Mach for the *Red* units. The safety constraints D_{unit} and D_{ground} are set to 2 km and 5 km, respectively. Before presenting the main results, a comparison of time consumption among all three schemes are conducted and its results are summarized in Figure 6(a), which demonstrates that the ES algorithm performs relatively faster only when the unit number is small, and as N increases, the proposed KM algorithm performs at a much faster pace than all other solutions, which is consistent with the complexity conclusion.

TABLE I: Initial conditions

Unit	Coordinates (km)	Weapons
BW_1	(720, 610, 20)	4
BW_2	(710, 620, 20)	4
BW_3	(720, 620, 20)	4
BW_4	(710, 610, 20)	4
BW_5	(700, 620, 20)	4
BB_1	(700, 600, 20)	5
BB_2	(800, 600, 20)	5
RD_1	(220, 210, 0)	3
RD_2	(210, 220, 0)	3
RD_3	(210, 200, 0)	3
RD_3	(210, 210, 0)	3
RD_5	(200, 210, 0)	3
RD_6	(200, 200, 0)	3
RD_7	(200, 220, 0)	3
FT	(200, 200, 0)	4

TABLE II: Probability of kill

	BW_1	BW_2	BW_3	BW_4	BW_5	BB_1	BB_2	RD_1	RD_2	RD_3	RD_4	RD_5	RD_6	RD_7	FT
BW_1	0	0	0	0	0	0	0	0.6	0.7	0.6	0.5	0.6	0.7	0.6	0
BW_2	0	0	0	0	0	0	0	0.6	0.6	0.5	0.7	0.6	0.7	0.6	0
BW_3	0	0	0	0	0	0	0	0.6	0.7	0.6	0.5	0.6	0.7	0.6	0
BW_4	0	0	0	0	0	0	0	0.6	0.6	0.5	0.7	0.6	0.7	0.6	0
BW_5	0	0	0	0	0	0	0	0.6	0.7	0.6	0.5	0.6	0.7	0.6	0
BB_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6
BB_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6
RD_1	0.4	0.5	0.4	0.6	0.4	0.5	0.5	0	0	0	0	0	0	0	0
RD_1	0.4	0.5	0.4	0.5	0.4	0.5	0.6	0	0	0	0	0	0	0	0
RD_1	0.4	0.5	0.4	0.5	0.4	0.5	0.5	0	0	0	0	0	0	0	0
RD_1	0.4	0.6	0.4	0.6	0.4	0.5	0.5	0	0	0	0	0	0	0	0
RD_1	0.4	0.5	0.4	0.5	0.4	0.5	0.5	0	0	0	0	0	0	0	0
RD_1	0.4	0.5	0.4	0.6	0.4	0.3	0.5	0	0	0	0	0	0	0	0
RD_1	0.4	0.5	0.4	0.6	0.4	0.5	0.5	0	0	0	0	0	0	0	0
FT	0.4	0.3	0.4	0.4	0.4	0.6	0.5	0	0	0	0	0	0	0	0

Figure 5 illustrates the combat trajectories between BW_1, RD_4 and RD_7, implying that when $k = 1850$ and $k = 3280$, BW_1 locked RD_4 and RD_7 as its targets, respectively, and both units exhibit barrel-roll maneuver, which is expected. Figure 6(b) shows the combat performance, where the number of weapons is normalized to be between 0 and 1. Noted that at the end of the simulation, the *Blue* force manages to inflict more than 50% of damage to the FT, signifying a victory but at the cost of heavy damage on the BB_1, BW_2 and BW_3, which attributes to the fact that the weighting coefficients of these damaged

TABLE III: Weighting coefficients

	B_1	B_2	B_3	B_4	B_5	B_6	B_7	R_1	R_2	R_3	R_4	R_5	R_6	R_7	FT
β^B	0.5	0.5	0.5	0.6	0.5	3	4	0.6	0.6	0.5	0.4	0.6	0.5	0.6	8
β^R	0.8	0.9	0.9	0.5	0.5	5	6	0.5	0.8	0.7	0.6	0.6	0.6	0.8	10

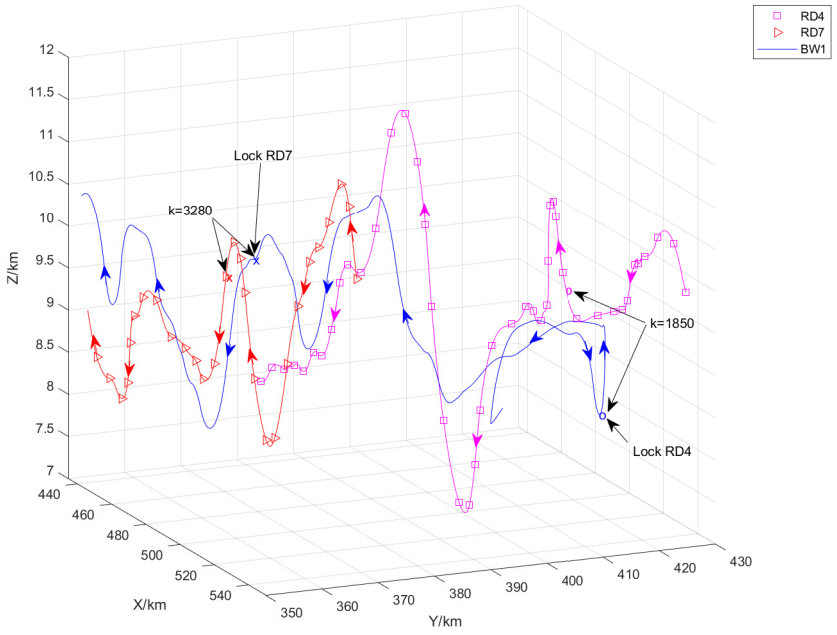
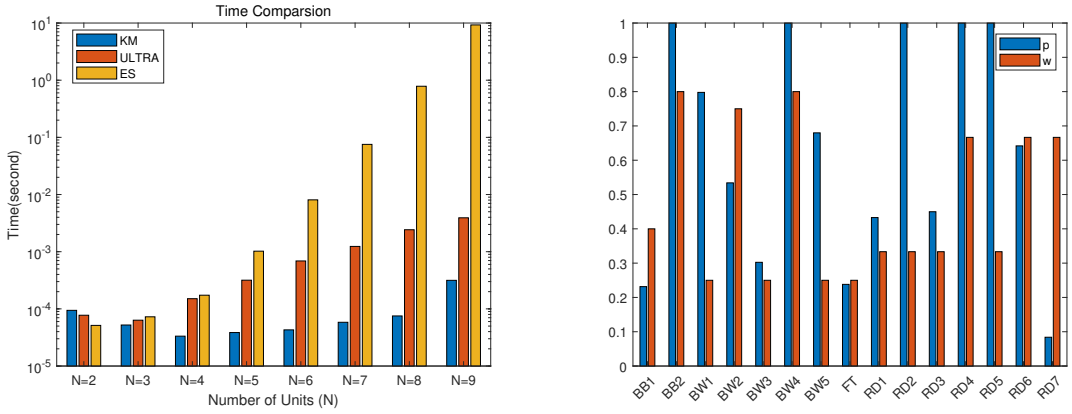


Fig. 5: Trajectories of BW_1, RD_4 and RD_7

units are higher than others, meaning that they are prioritized over others for the *Red* forces. On the other hand, the *Red* force suffers serious losses as well, almost half of its units is in severe condition.



(a) Time consumption

(b) Combat performance

Fig. 6: Simulation results

V. Conclusions

This paper investigated task management and control problem for military operations, we demonstrated that the underlying problem can be converted to a maximum weight matching problem associated with a bipartite graph, and we proposed an extension of the Kuhn-Munkres algorithm to search for an optimal response strategy with action-reaction strategy. Simulation

results show the one-step look-ahead Nash solutions can be achieved for both sides, and application of maximum weight matching in task management problem can considerably reduce its algorithmic complexity, and consequently makes real-time decision possible.

ACKNOWLEDGMENTS

This work is supported in part by the Zhejiang Provincial Natural Science Foundation of China (Grant No. LR20F030003), and by the National Natural Science Foundation of China (Grant No. 62088101 and 91748128).

REFERENCES

[1] K. Virtanen, J. Karelahti, and T. Raivio, "Modeling air combat by a moving horizon influence diagram game," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 5, pp. 1080–1091, 2006.

[2] J. B. Cruz, M. A. Simaan, G. Aca, H. Jiang, B. Letellier, M. Li, and Y. Liu, "Game-theoretic modeling and control of a military air operation," *IEEE Transactions on Aerospace & Electronic Systems*, vol. 37, no. 4, pp. 1393–1405, 2001.

[3] J. Cruz, J. B., M. A. Simaan, A. Gacic, and Y. Liu, "Moving horizon Nash strategies for a military air operation," *IEEE Transactions on Aerospace & Electronic Systems*, vol. 38, no. 3, pp. 989–999, 2002.

[4] Y. Liu, M. A. Simaan, and J. B. Cruz, "An application of dynamic Nash task assignment strategies to multi-team military air operations," *Automatica*, vol. 39, no. 8, pp. 1469–1478, 2003.

[5] X. Chen and Y. Hu, "Study on the task assignment for multi-UCAV in air-combat," in *2012 Fifth International Symposium on Computational Intelligence and Design*, vol. 1, 2012, pp. 448–451.

[6] X. Chen, T. Tang, and L. Li, "Study on the real-time task assignment of multi-UCAV in uncertain environments based on genetic algorithm," in *IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, 2011, pp. 73–76.

[7] L. Zhong, Q. Luo, D. Wen, S. Qiao, J. Shi, and W. Zhang, "A task assignment algorithm for multiple aerial vehicles to attack targets with dynamic values," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 236–248, 2013.

[8] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. SIAM, 1998.

[9] J. F. Nash, "Equilibrium points in N-person games," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 36, no. 1, pp. 48–49, 1950.

[10] Y. Chai, J. Luo, N. Han, and J. Sun, "Robust event-triggered game-based attitude control for on-orbit assembly," *Aerospace Science and Technology*, vol. 103, p. 105894, 2020.

[11] N. Han, J. Luo, Z. Zheng, and J. Sun, "Distributed cooperative game method for attitude takeover of failed satellites using nanosatellites," *Aerospace Science and Technology*, vol. 106, p. 106151, 2020.

[12] D. Ye, M. Shi, and Z. Sun, "Satellite proximate pursuit-evasion game with different thrust configurations," *Aerospace Science and Technology*, vol. 99, p. 105715, 2020.

[13] J. Li, S. Chen, C. Li, and F. Wang, "Distributed game strategy for formation flying of multiple spacecraft with disturbance rejection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 1, pp. 119–128, 2021.

[14] H. Liang, J. Wang, J. Liu, and P. Liu, "Guidance strategies for interceptor against active defense spacecraft in two-on-two engagement," *Aerospace Science and Technology*, vol. 96, p. 105529, 2020.

[15] X. Chen, X. Deng, and S.-H. Teng, "Settling the complexity of computing two-player Nash equilibria," *Journal of the ACM*, vol. 56, no. 3, pp. 1–57, 2009.

[16] D. Galati and M. Simaan, "Effectiveness of the Nash strategies in competitive multi-team target assignment problems," *IEEE Transactions on Aerospace & Electronic Systems*, vol. 43, pp. 126 – 134, 2007.

[17] D. Galati, Y. Liu, and M. A. Simaan, "A fast algorithm for unit level team resource allocation in a game environment," in *IEEE Conference on Decision and Control*, vol. 3, 2003, pp. 2872–2877.

[18] H. Duan, P. Li, and Y. Yu, "A predator-prey particle swarm optimization approach to multiple UCAV air combat modeled by dynamic game theory," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 1, pp. 11–18, 2015.

[19] S. Orafa, M. J. Yazdanpanah, C. Lucas, A. Rahimikian, and M. Nili Ahmadabadi, "Development of reinforcement learning methods in control and decision making in the large scale dynamic game environments," in *IEEE Conference on Computer Aided Control System Design, IEEE International Conference on Control Applications, IEEE International Symposium on Intelligent Control*, 2006, pp. 850–855.

[20] J. S. Przemieniecki, *Mathematical Methods in Defense Analyses*. AIAA Education Series, 2000.

[21] A. W. Starr and Y. C. Ho, "Nonzero-sum differential games," *Journal of Optimization Theory and Applications*, vol. 3, no. 3, pp. 184–206, 1969.

[22] W. Lin, C. Li, Z. Qu, and M. A. Simaan, "Distributed formation control with open-loop Nash strategy," *Automatica*, vol. 106, pp. 266–273, 2019.

[23] G. Chartrand, *Introductory Graph Theory*. Courier Corporation, 1977.

[24] L. R. Ford Jr and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 2015.

[25] D. G. Galati, "Game theoretic target assignment strategies in competitive multi-team systems," Ph.D. dissertation, University of Pittsburgh, 2005.
[26] L. Li, Z. Li, Y. Li, B. Kathariya, and S. Bhattacharyya, "Incremental deep neural network pruning based on Hessian approximation," in *Data Compression Conference*, 2019, pp. 590–590.
[27] D. E. Drake and S. Hougardy, "A simple approximation algorithm for the weighted matching problem," *Information Processing Letters*, vol. 85, no. 4, pp. 211 – 213, 2003.
[28] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of The Society for Industrial and Applied Mathematics*, vol. 10, pp. 196–210, 1957.
[29] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.
[30] ——, "Variants of the Hungarian method for assignment problems," *Naval Research Logistics Quarterly*, vol. 3, no. 4, pp. 253–258, 1956.